Stony Brook University

Data Discovery, Caching, Security and Architecture Support in Edge Computing

Yuanyuan Yang

Distinguished Professor of Electrical and Computing Engineering

and Computer Science

Stony Brook University

National Science Foundation

FAR BEYOND

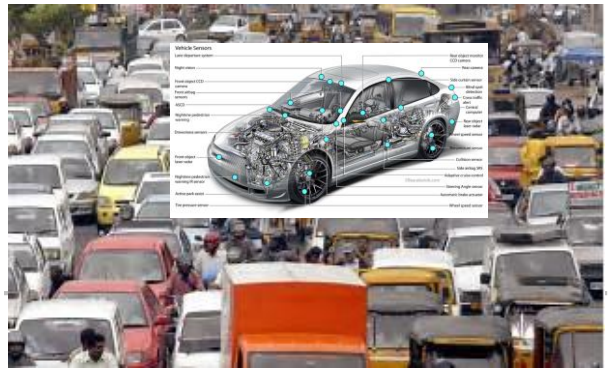
Outline

- Pervasive edge computing (PEC)
  - Motivation and concept
  - Challenges, opportunities and solution
- Data discovery and retrieval in PEC
- Data caching in PEC
- Blockchain for secure data caching in PEC
- Architecture support for PEC
- Future research agenda in PEC
- Summary

Stony Brook University

Emerging Edge Devices

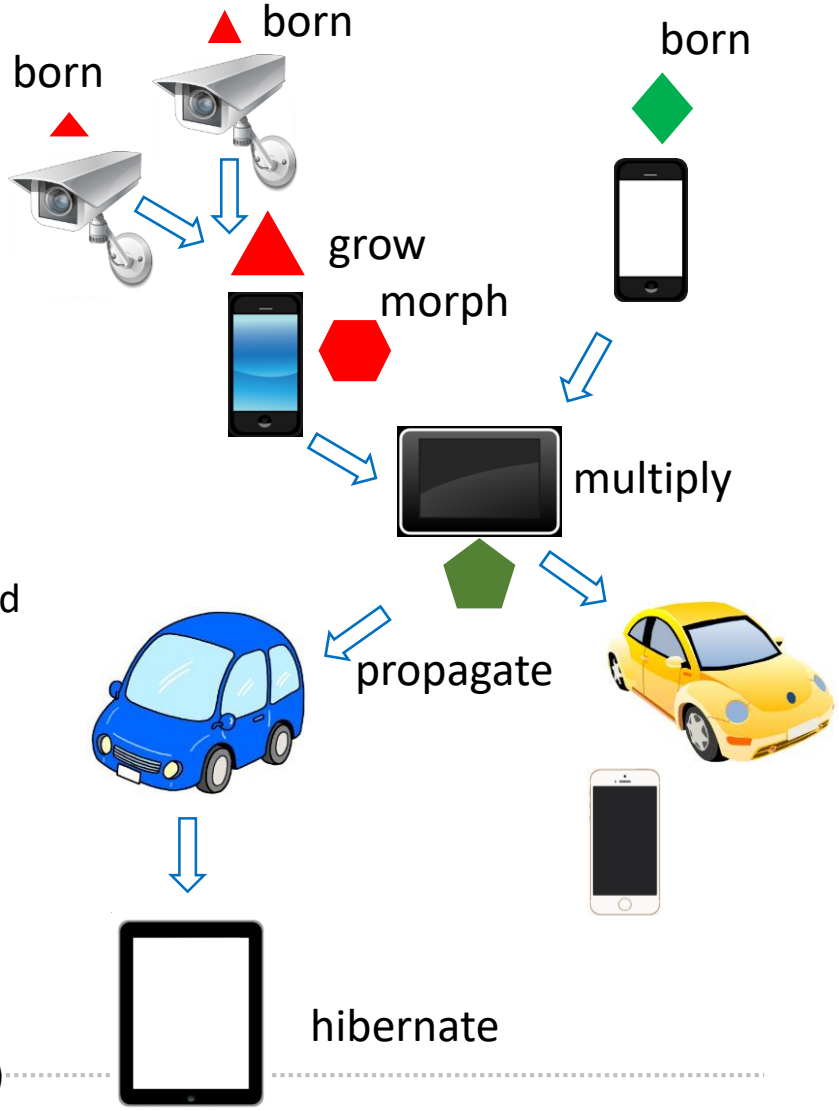
- Various devices of rich, diverse resources
  - phones/tablets/wearables, IoT, vehicles
  - perform sensing, communication, storage, processing operations
- Pervasive, high density distribution everywhere by opportunistic congregation
  - customers in cafes, passengers at boarding gates
  - IoT devices in homes, offices, shopping malls
  - vehicles around city streets



Stony Brook University

New Computing Paradigm: Pervasive Edge Computing (PEC)

- Data storage /processing and decision all are performed at the edge
  - Contrast to cloud computing where the edge provides only data, but storage, processing and decision are done in the backend
- Promote data as **first class** citizens
  - Data become **independent** entities decoupled from specific devices (e.g., their producers)
  - Same/similar data/resources can be provided by different devices
    - Whichever willing/capable should be used
- Data become **living** organisms
  - born (gathered by sensors)
  - grow (accumulated over space/time)
  - morph (transformed in syntax/semantics after processing)
  - multiply (information extracted by merging different data inputs)
  - propagate (disseminate)
  - eventually hibernate/die (archived/eliminated)



FAR BEYOND

Advantages of PEC

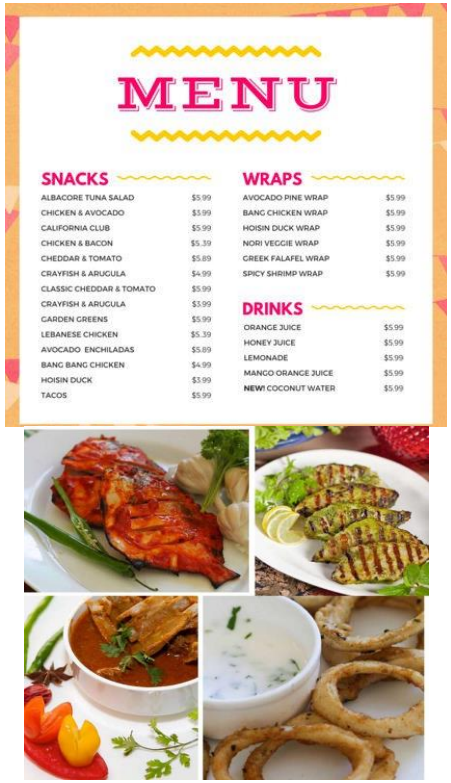
- Zero barrier
  - No infrastructure cost/complexity in deploying, maintaining dedicated cloud backend
- Low latency
  - No need to communicate with cloud. Processing occurs near data, greatly reducing back-and-forth transmission delay
- Connectivity independence
  - No internet necessary. Communication among peer devices even without the Internet
- Democratic creativity
  - No central authority. Devices are free to decide with which others to collaborate and how, enabling flexible and diverse applications

Challenges and Opportunities in PEC

- Challenges
  - Heterogeneity: types/quantities of device capability/resources vary greatly
  - Uncertainty: highly dynamic changes in mobility, connectivity, resource availability (sensing/computing/storage/communication)
  - Diverse ownership: no central authority, mostly voluntary sharing
- Opportunities
  - Pervasive distribution: devices everywhere
  - Existence of more powerful devices: vehicles, main powered, to help coordinate peers and perform intensive work for others

Peer Data Sharing (PDS)

- Discover nearby, opportunistically available data, and retrieve them
  - avoid blindly requesting non-existent data
- Peer Data Discovery (PDD)
  - Build a “menu” of available data
- Peer Data Retrieval (PDR)
  - Retrieve desired data



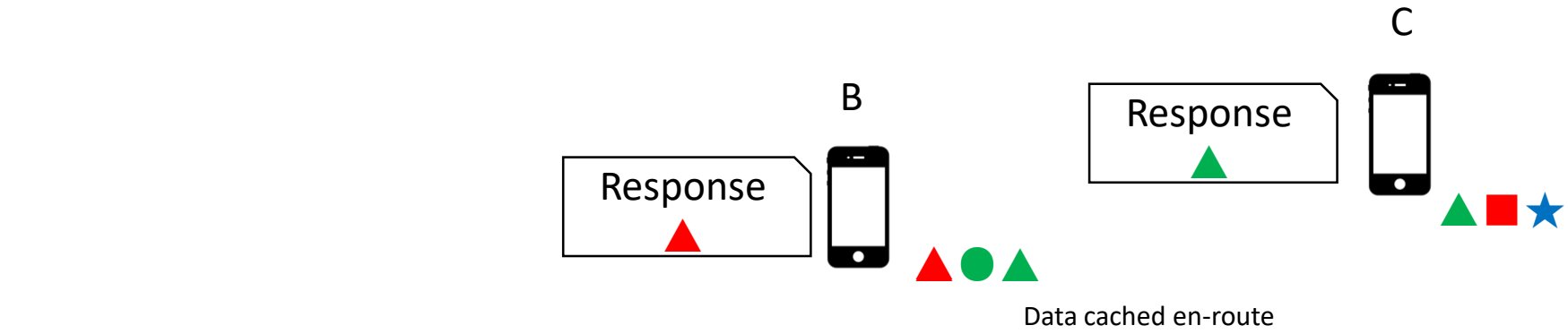
Data as Independent Entities

- Data are self-sufficient entities decoupled from original producers
  - widely cached, thus referenced, stored and accessed from any willing and capable devices
- Data Descriptors
  - When produced, each data item is associated with a **data descriptor** (a metadata entry)
    - Metadata is also a kind of data
      - Widely cached, possibly separately from respective data
    - All such entries together describe what data *may* exist in the environment (the menu)
  - One possible form: key-value pairs
    - {key: value}
    - Example
      - Data Type: **NO_2** sample
      - Attributes: **time** and l**ocation** of sample
      - Namespace: **environment monitoring/atmosphere**

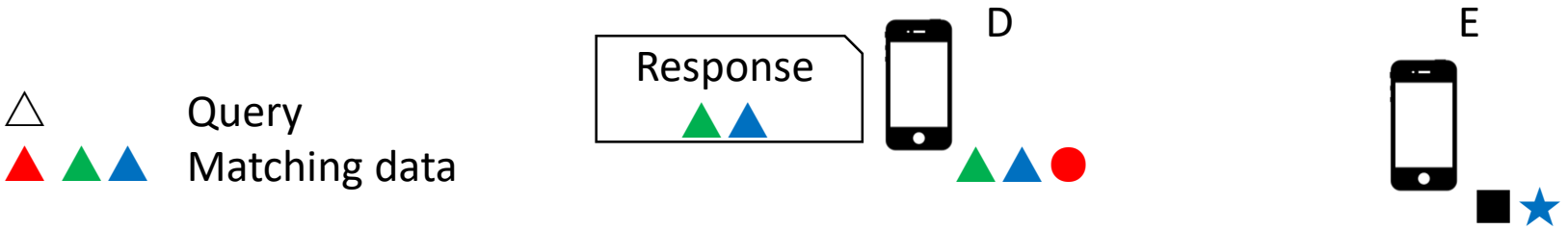
Data Query, Response and Caching

- A consumer sends **queries** to specify desired data and retrieve them from other devices
- Queries propagate over the network towards desired data
  - Flooded if no routing information for the data exists (e.g., newly generated data)
  - Otherwise, follow content-centric routing to reach the data (explained later in PDR)
- Response
  - A provider sends **responses** to return data that are requested by queries
  - Responses are delivered back to the consumer following the reverse path of the query
- Caching
  - Intermediate nodes can cache the data for future requests
    - Shorter access latency for future requests
    - Better robustness to uncertainty (e.g., node mobility, connectivity failure)

Query, Response and Caching



1. The consumer propagates a query
2. Each device includes matching data in its response
3. Responses follow reverse path to reach the consumer, and their data may be cached en-route



Peer Data Retrieval (PDR)

- Once the "menu" of data descriptors is retrieved, the consumer decides which data items to "order"
- Two typical scenarios: large and small data
  - Scenario 1: a large data item (e.g., a video clip) consisting of many small **chunks**, each cached by multiple devices.
  - Scenario 2: many small data items cached by multiple devices (e.g., air pollution samples in certain area).
- We focus on Scenario 1. Scenario 2: follow the same process as metadata discovery discussed earlier
- **Question: How to obtain a copy of each chunk quickly and efficiently?**

Two-Phase Data Retrieval

- Phase 1: Building content-centric routing tables
- Phase 2: Recursive chunk retrieval

- Content-centric routing table
  - Each routing entry at a node {data_chunk_ID, m, n_{ID}}

    tells how far in minimum hops **m**, via which neighbor(s) **n_{ID}** to reach a chunk **data_chunk_ID**

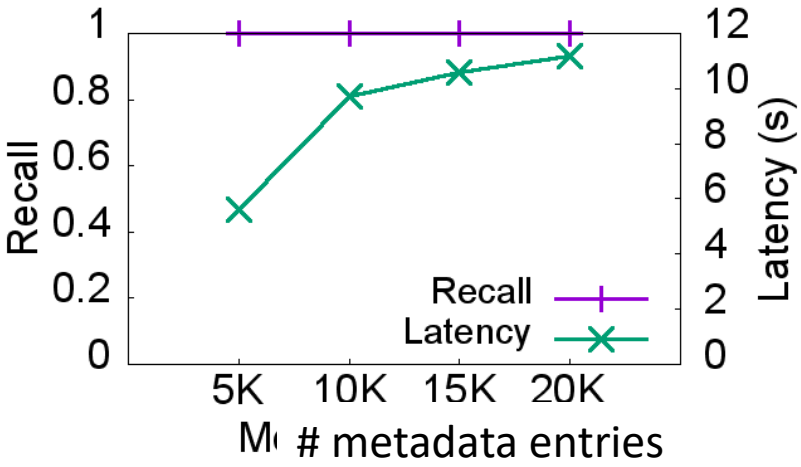
Creating Routing Tables

- Phase 1: Routing Table Creation
  - A consumer propagates a query for routing table entries of desired data D
  - A node M_0 with desired chunk i responds: {D-i, 0, -}
  - At hop k, node M_k adds a local routing entry: {D-i, k, M_{k-1}}, and continues forwarding this entry towards the consumer
  - Finally at the consumer: via which neighbor, in how many hops to reach the nearest copy for each chunk

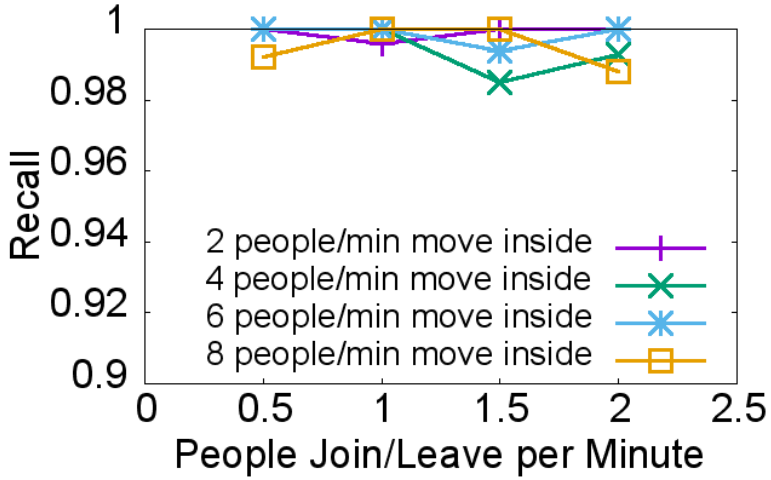
Recursive Chunk Retrieval

- Phase 2: Recursive Chunk Retrieval
  - A consumer **sends** sub-queries, each requesting a subset of chunks from a **direct neighbor**
  - Upon receiving a sub-query, a node
    - **replies** requested **chunks** it has
    - **divides** remaining chunks into sub-queries, then **sends** to neighbors with minimum hops to those chunks
  - Upon receiving replied chunk, a node **forwards** it back towards the **consumer**
  - Chunks are retrieved recursively from all downstream nodes
- Use an algorithm to balance the load among neighbors to shorten retrieval latency
  - Some neighbors on shortest paths send more chunks, thus more workload
  - Objective: minimize the maximum load among all neighbors
  - First assign each chunk to a neighbor with smallest hop count
  - Heuristic: move a chunk from the neighbor with highest load to the neighbor with the 2nd shortest path; repeat until the highest load no longer decreases

Data Discovery Completeness and Latency



- 100% discovery in a few seconds



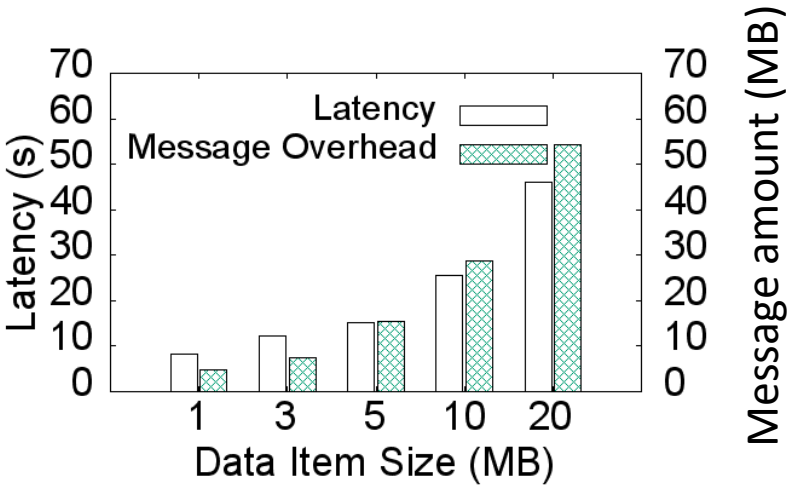
- 100% discovery despite mobility

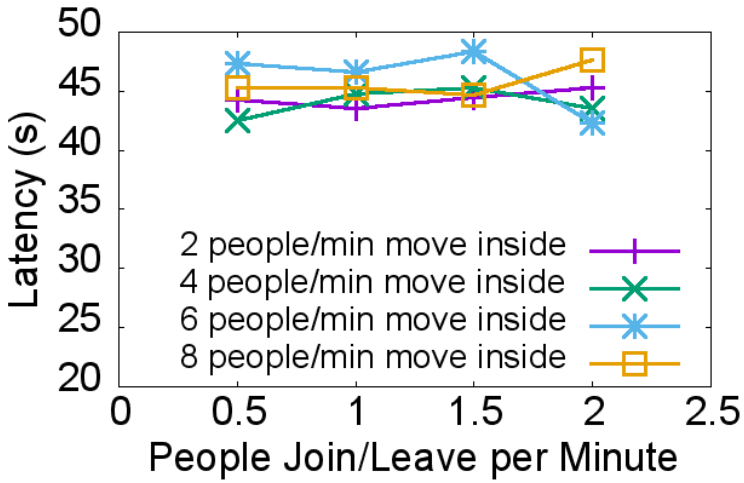
Observation: PDD discovers all metadata entries in short time despite mobility

Setup:

- 100 node network
- Simulation parameters based on an Android prototype
- Mobility based on 8 hours of real world human movement observation

Data Retrieval Latency





- 10's of seconds for 10-20MB data;
- Retrieve nearest copies in 2-3 hops (message amount 2-3 times of data size)

- Latency unchanged despite mobility

Observation: PDR retrieves large size data in short time

Data Caching

- In PEC, a node stores a copy of data it forwards, overhears, or receives
  - The node itself may not need the data
- Benefits
  - Cached copies can be used to meet consumer needs
  - Less latency: nearest copy sent
  - More robustness: copies exist after some node failure/mobility
  - Higher bandwidth: different portions of the same data can be retrieved from multiple copies
- Question: **Which node should store a copy of data**?

Limitation of Existing Caching Work

- Fairness in workload is not considered
  - Assume all nodes obey one authority (e.g., owner of the whole network)
  - Some ``best'' nodes are dictated to shoulder more caching workload than others
  - Node ability and willingness are not considered
- In edge computing, no central authority exists
  - devices belong to different owners; each has limited resources and wants to control its resource contribution

Metric for Caching Fairness

- Fairness Degree Cost
  - The less resources, the higher the cost, thus less willing to cache more data
  - One simple form
    - $f = \frac{storage\ used}{storage\ available}$
- Intuition: put data on nodes with more available resources, thus lower cost

Metric for Contention Cost

- Wireless contention causes delays in data access
  - Collisions due to multiple nodes send simultaneously, thus loss of messages
  - Back-off time, retransmission add to the delay
- Contention induced-delay cost
  - Quantify contentions along data delivery path between nodes i and j
  - Proportional to total amount of data messages along the path

Cache Placement Problem Formulation

- Problem: Given a network topology, for each data chunk, determine which nodes should cache a copy
- Objective: Minimize a weighted sum of fairness and contention costs
- Placement should achieve **fairness** (similar loads among nodes) while minimizing contention (thus retrieval **latency**)
- An integer programing problem is formulated
  - Map to multiple Connected Facility Location (ConFL) problems added together
  - ConFL problem is an NP-hard problem, thus, our problem is also NP-hard

Centralized Approximation Algorithm

- The algorithm adopts a 6.55-approximation algorithm of ConFL problem as a subroutine
- It preserves the approximation ratio
- Complexity is $O(N^3)$, *N:* number of nodes in the network
- Limitation: require global information at each node

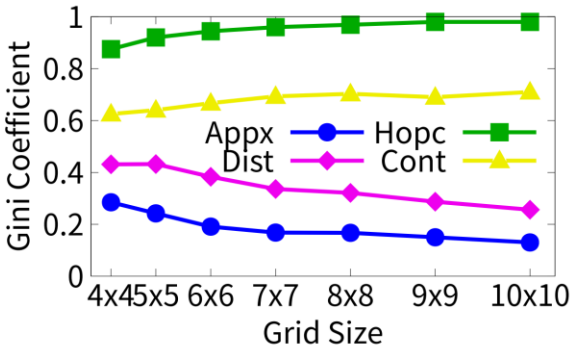
Extend to Distributed Algorithm

- Each node needs to decide individually whether or not to cache
  - The centralized approximation algorithm requires global information, which is difficult to obtain
- Basic idea of distributed algorithm
  - Each node asks other nodes within k hops whether they can help cache the data
    - k is a preset parameter, e.g., 2-3
  - If a node receives more `yes' responses, it is more likely to cache data
- Message overhead: $O(QN + N^2)$
  - Q: number of data chunks cached
  - N: number of nodes

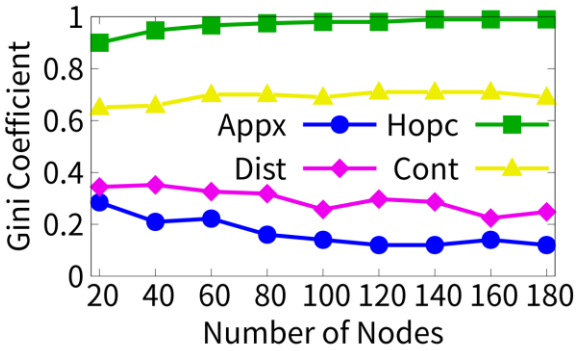
Stony Brook University

Fairness Evaluation

- Compare with two existing algorithms:
  - Contention based (Cont): minimizing total contention cost
  - Hop count based (Hopc): minimizing total transmitted hops
- Gini coefficient is a common metric to quantify (in)equality in a set of values
  - Value between 0-1: the smaller, the better (fairer)



(a) Grid network



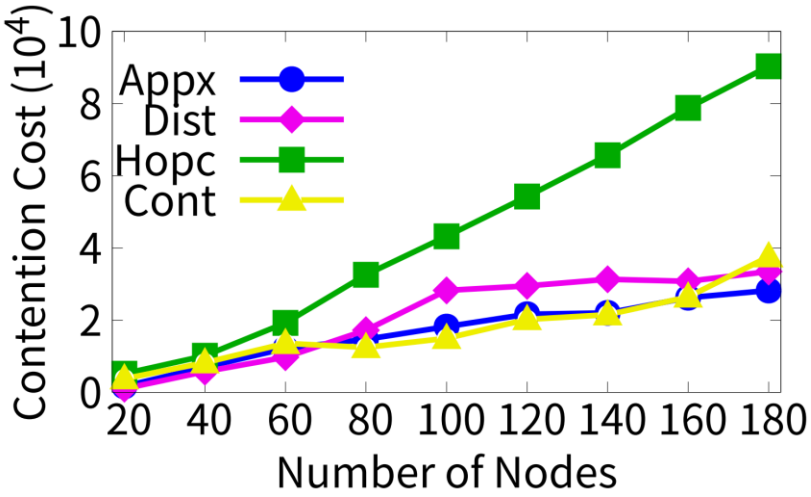
(b) Random network

- Our algorithms (Appx/Dist) achieve much better fairness than two existing algorithms (Cont/Hopc)

FAR BEYOND

Contention Cost Evaluation

- Contention-induced delay cost on random networks



- Minimum contention cost
  - Our algorithms are comparable to Cont, which has minimum contention cost
  - Our algorithms are much lower than Hopc
- **Observation**: our algorithms achieve minimum contention cost while greatly improving fairness

Secure Data Caching in PEC

- A major concern in edge computing: **security**
  - Data transactions and privacy protection in an edge computing environment often have to involve a trusted third-party
- **Blockchain** is a promising technology to combat this challenge. It can be used to ensure data transactions unmodifiable and undeniable
  - A blockchain system consists of a chain of blocks (records). Each block contains the hash result from its previous block to form a chain
  - Blockchain has many security features over a regular distributed system
    - keep complete transaction history
    - designed against data modification
    - No need for a trusted third-party in the transactions and can avoid “a central point of failure”

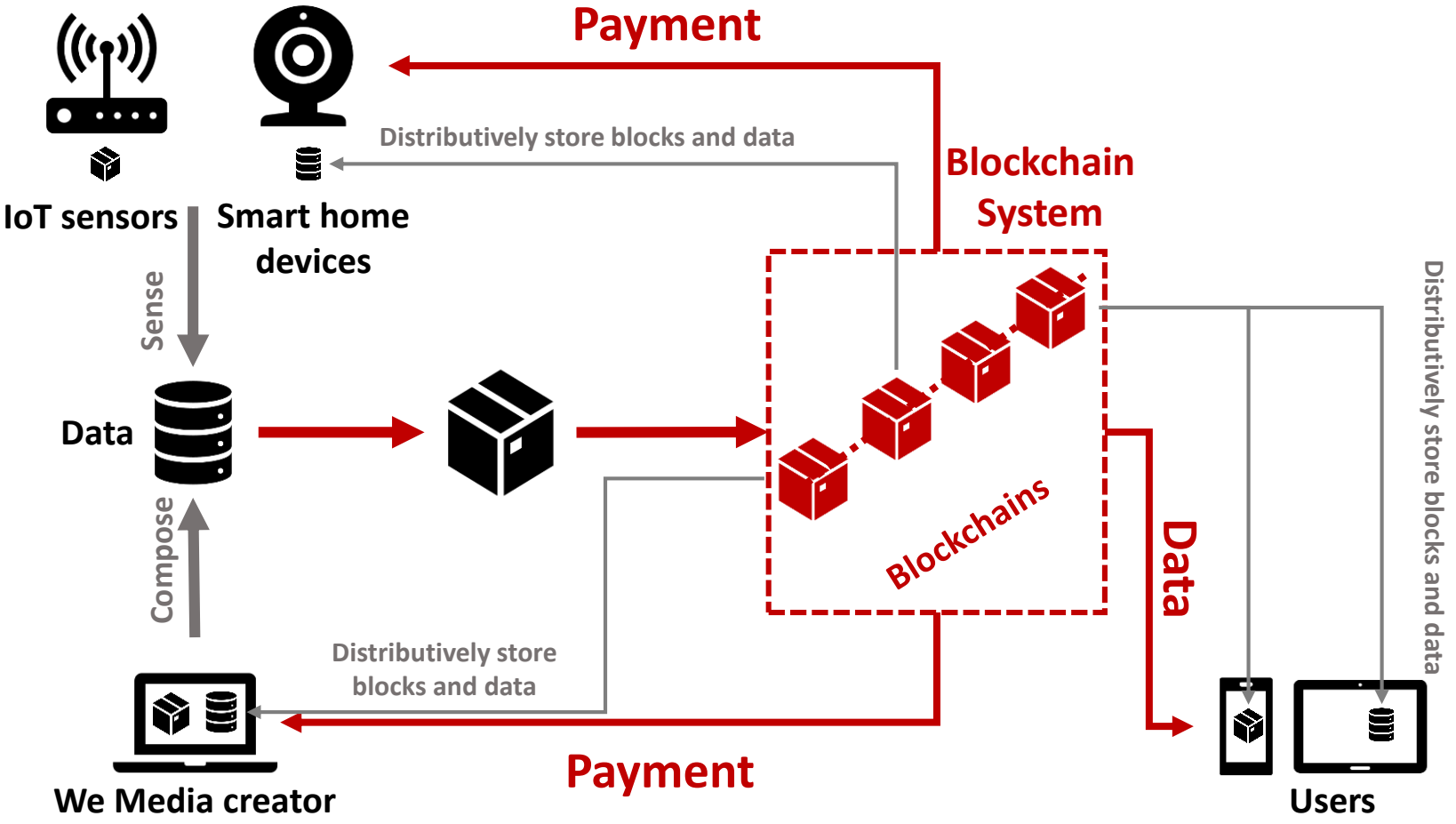
Blockchain on Edge

- Combining blockchain with edge computing can have many advantages that ensure
  - **Secure payment**: Every transaction is recorded in the blocks, thus cannot be changed or denied
  - **Access-control:** Data producers can directly manage subscriptions to deliver for-profit data to consumers
  - **Privacy**: Blockchain can protect the identity of the users
- Challenges
  - Limited resources, e.g., storage, computing power
  - Resource constraints of edge devices reduce the applicability of blockchain
    - Storing all blocks and data on every edge node is not feasible
    - Edge devices have less computing power and energy capacity compared to traditional blockchain miners
- Directly applying the existing blockchain approach onto edge is not feasible

Blockchain on Edge - Our Approach

- Design a blockchain system suitable for edge environment, focusing on reducing storage, latency and energy cost
  - Instead directly storing data in blocks, only store metadata in blocks, and distributively store data and blocks in key locations for quick access
    - Data can be searched through metadata, and can be accessed from nearby cached nodes
    - Blocks are also distributively stored among nodes
  - Adopt **Proof of Stake (PoS)** consensus mechanism to achieve consensus among nodes to reduce energy cost
    - It does not require a compute-intensive process to achieve consensus as **Proof of Work (PoW)** consensus mechanism

Blockchain on Edge - System



Fair and Efficient Storage Distribution

- After metadata are created and stored in blocks, data items are then stored onto nodes distributively
  - Metadata contain where the data item is stored and users can access the data based on the information
- Some nodes in critical locations can be accessed easily. Such nodes will store data items for other nodes to access
- Fair (balanced) storage enables more nodes to participate in caching and avoids overloads
  - Nodes with more resources and at key locations are more likely to be selected to store data and blocks

Recent Block Storage

- Edge devices, often with wireless transmission and mobility, may miss newly generated data and blocks
- Recently accessed blocks are more likely to be requested and more vulnerable to temporary disconnection of nodes
  - If recent blocks are more pervasive in the network, it is easier to retrieve them
- Nodes are required to cache a certain number of most recent blocks and replace the blocks using FIFO
  - The number of recent blocks a node caches also depends on the locations and fairness

Consensus Mechanism (PoW)

- A blockchain system needs a consensus mechanism to achieve distributed consensus on new block generation

Proof of Work is a commonly used consensus mechanism, e.g., Bitcoin

- Miners compete to find a certain value, when hashed together with the whole block, the hashed value satisfies a certain target value
- Very time and energy consuming process
- Relatively secure, if more than half of nodes in the system are honest

Consensus Mechanism (PoS)

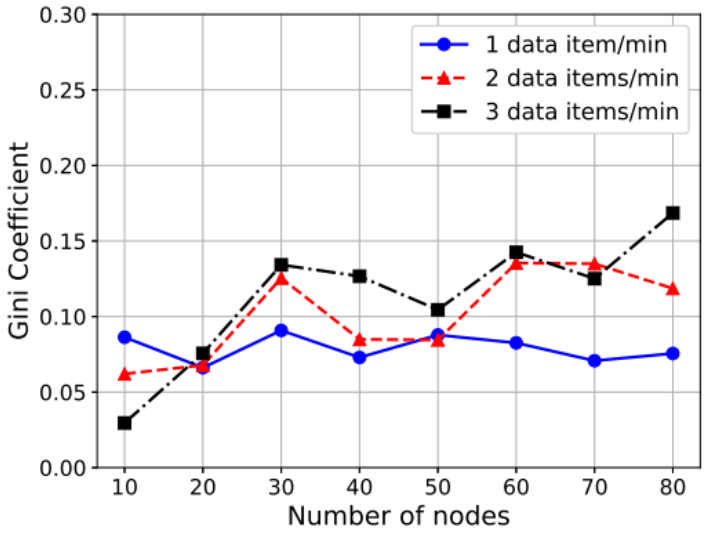
Proof of Stake (PoS) is another commonly used consensus mechanism

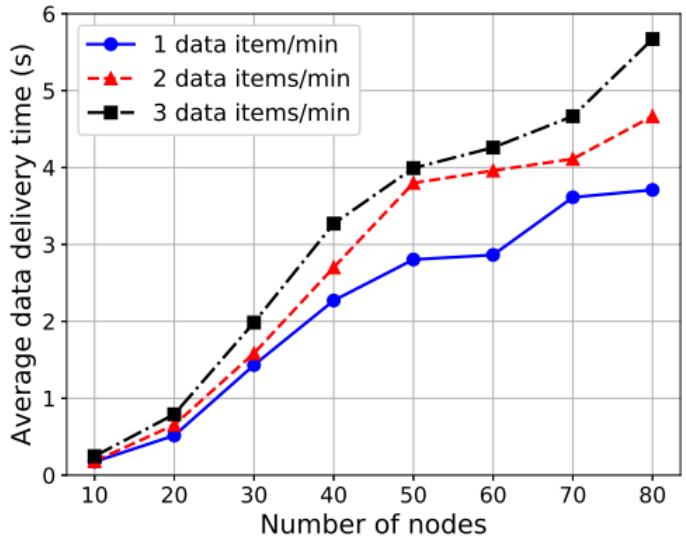
- Some cryptocurrencies have adopted this idea, e.g., Peercoin
- Creator of the next block is randomly selected based on the stake, e.g., wealth, age, contributions
- Less time for devices to compute and reach consensus
- Maybe less secure compared to PoW
- More practical to implement PoS in edge environment

Proof of Stake

- **Selection rule**: If a node contributes more, i.e., storing more data and creating more blocks, it will have a higher probability than others to be the creator of the next new block
- We reward the node that has contributed more to the system

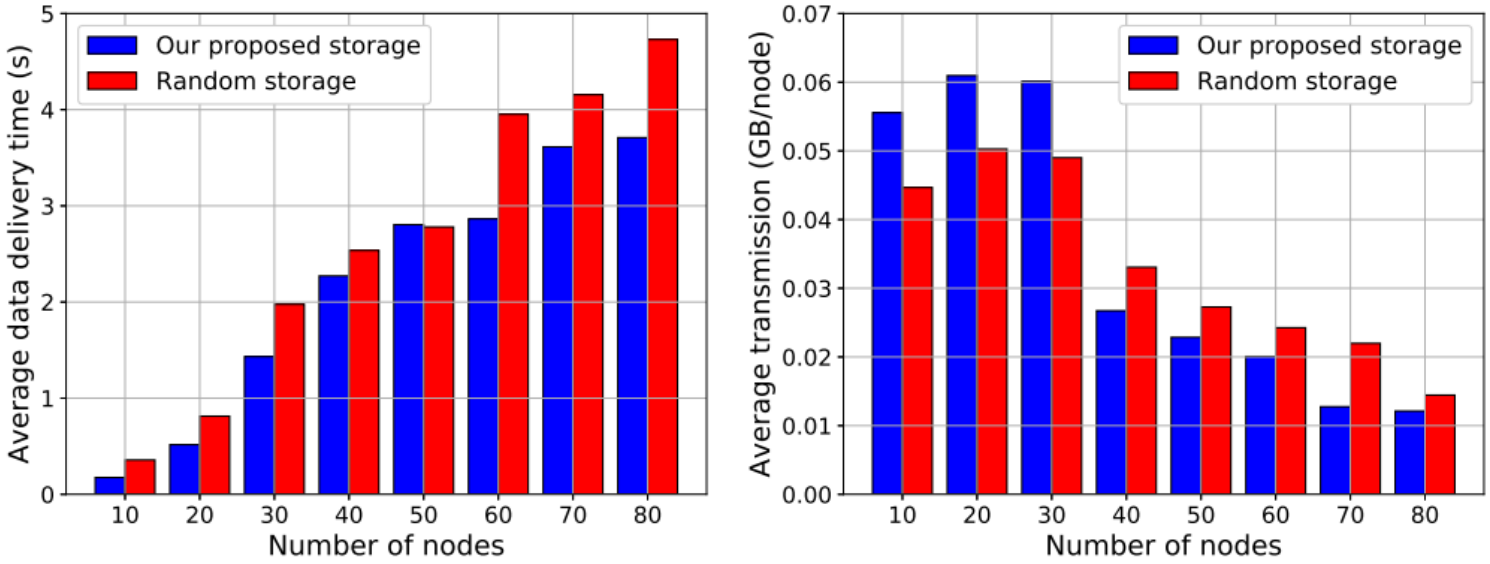
Performance under Different Data Generating Rates





- 1-3 data items generated per minute, in total 500 blocks generated
- Gini coefficient (left) for date storage (load balance) on nodes and average delivery time (right) for different network sizes
- For Gini coefficient, it remains very low among 0.15, showing small disparities among node storage (balanced load)
- Data delivery time grows almost linearly with the number of nodes, and remains almost the same with different data generating rates

Performance under Different Storage Allocation Strategies



- One data item generated per minute, running 500 blocks
- Average delivery time (left) and transmission overhead (right) over different network sizes
- For delivery time, the proposed data storage is shorter than random storage
- The average transmission amount is less when network size is large.

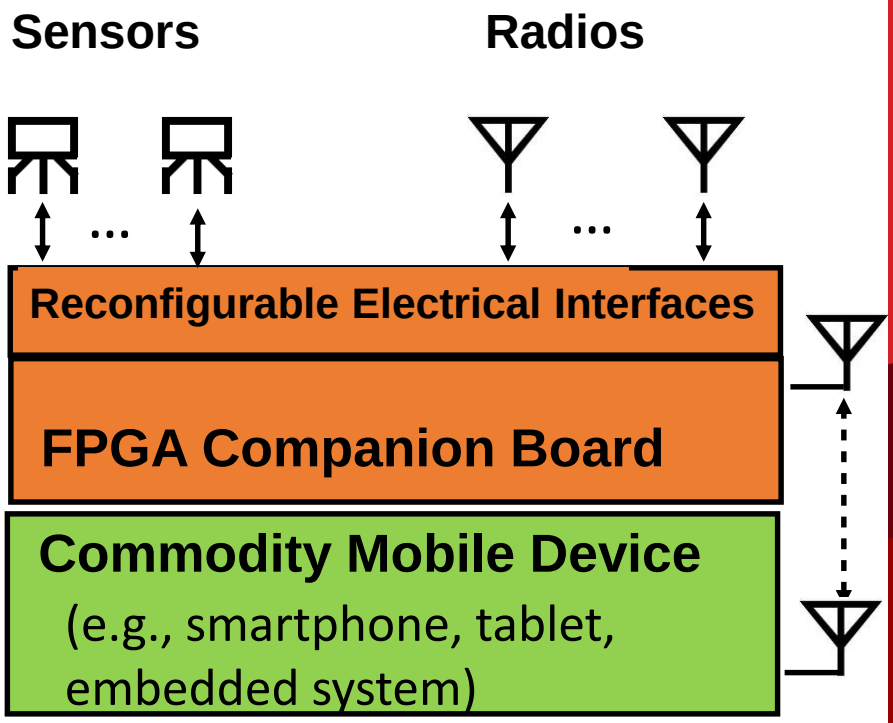
Architecture Support for PEC

- Edge devices exhibit huge heterogeneity: types/quantities of device capability/resources vary greatly
- Many edge applications demand both hardware and software specialization not available on commodity edge devices
- A modular, composable hardware/software architecture for customizable edge devices
  - Diverse modularized FPGA/software computation components
  - Easily composed electronically and computationally like Lego pieces
  - As a companion to augment a commodity mobile device like GPU in a multicore processor

Edge Node Architecture: Towards Low-Cost Modular Specialization

FPGA Accelerator Library
- data/signal processing
- security

Software Computation Component Library

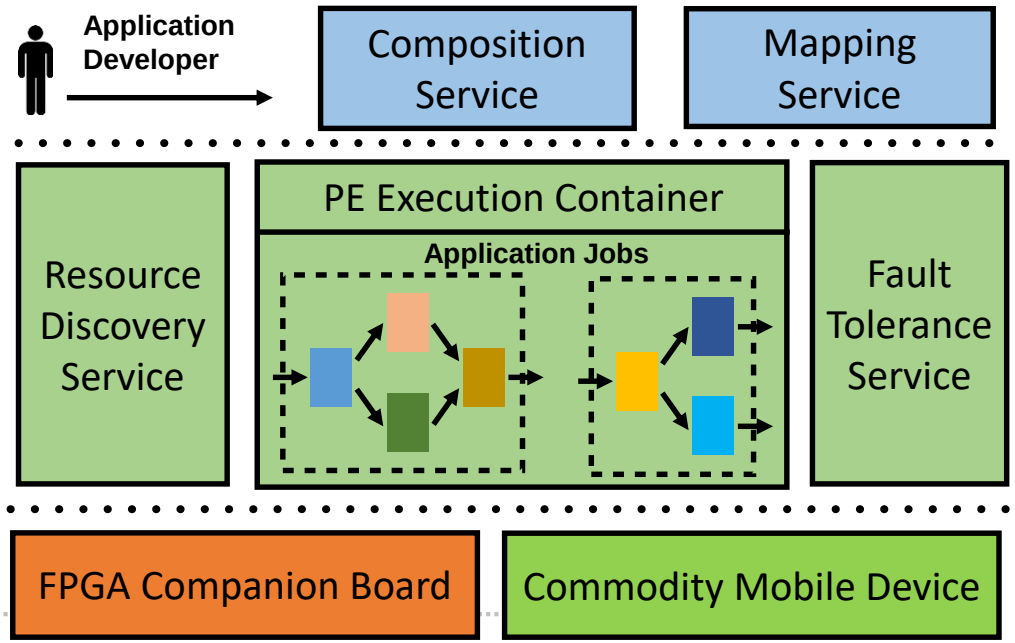


Stony Brook University

Data Stream Processing System Architecture on Customizable Edge Devices

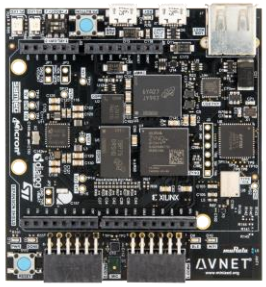
- A data processing system runs on customizable edge devices
  - **Resource discovery service** finds what hardware/software resources are available in local and nearby edge devices
  - **Composition service** creates jobs in form of connected graphs among modules, using discovered resources
  - **Mapping service** schedules which portions of the graph run on which edge nodes
  - **PE execution container** runs locally deployed application jobs
  - **Fault tolerance service** handles resource variability due to competition/mobility

- Developers compose reusable hardware/software modules and components into applications
- One application may require collaboration and run jobs among multiple edge devices

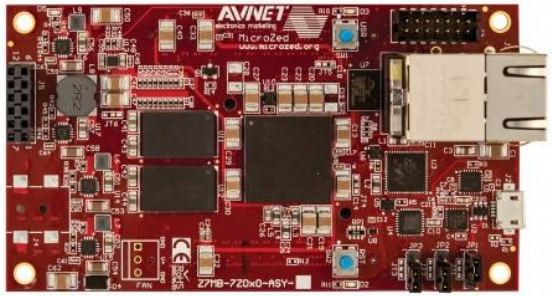


FAR BEYOND

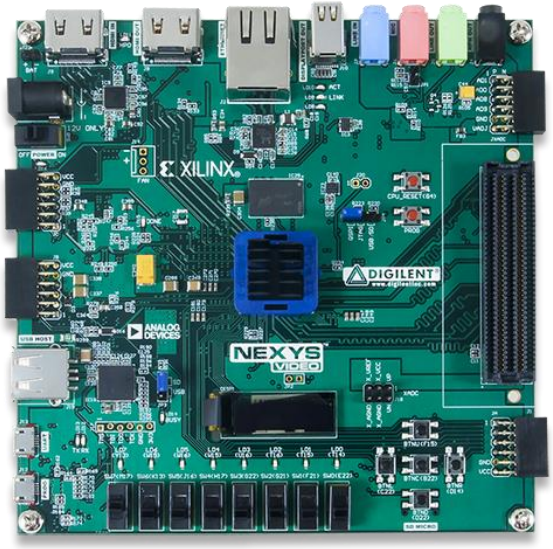
Three Types of “Pebble” Nodes



Socket grade for IoT: Zynq Z-7007S



Phone grade for mobile devices: MicroZed



Tablet grade for vehicles, edge servers:
Nexys Video Artix-7 FPGA

Future Research Agenda in PEC

- Discovery and retrieval at larger scale (e.g., campus)
  - Pure peer based is not scalable
  - Need the help of more powerful nodes
  - Build an infrastructure of more powerful nodes (e.g., static, more resources). They exchange information to store/serve metadata and data for others
- Cache management
  - Aggressive caching: cache anything overheard
    - Reason: storage is relatively abundant while contact is rare, esp. with mobility (e.g., vehicles)
  - How to replace existing data when cache is full
  - Incorprate emerging storage devices
- Autonomous, efficient processing under uncertainty (mobility, connectivity, resources)
  - Nodes automatically decide what processing to invoke where, on what raw data, to produce desired output

Stony Brook University

Future Research Agenda in PEC

- Trust management
  - How much trust on whose/what data, on trusted data processed by less trusted nodes, or merged output from producers of different trust levels?
- Security and privacy
  - How to protect against attacks?
    - Attacks are much easier to launch in a loosely coupled environment , like PEC, without central authority or definitive boundary
    - Such as injecting bogus data, claiming others' identities, …
  - How to confine the propagation scope of sensitive data?
- Incentive mechanisms
  - Participation is voluntary. How to compensate contributors (sensing/processing) appropriately to stimulate collaboration?
- Expressive naming structures
  - Adapt semantic web, linked data techniques for more expressive data descriptions based on content, not format
- Smarter edge: machine learning on edge, 5G and 6G devices

Summary

- A new paradigm of pervasive edge computing is emerging
  - Sensing, processing/decision all are done at the edge
  - Sharp contrast to cloud computing
- A data centric architecture offers a viable solution
  - Data as first class citizens
  - Robust and fast data discovery and retrieval
  - Fair cache placement
- Time for exciting research: the field is wide open
  - Architecture support
  - Scalable data discovery, retrieval
  - Cache management
  - Autonomous processing
  - Trust, security and privacy
  - Incentives
  - Semantic data naming
  - Machine learning on edge
  - 5G, 6G edge devices

Stony Brook University

Related Publications

- X. Song, Y. Huang, Q. Zhou, F. Ye, Y. Yang and X. Li, ``Pervasive Edge Data Sharing in MANET,'' IEEE INFOCOM/IECCO 2017.
- X. Song, Y. Huang, Q. Zhou, F. Ye, Y. Yang and X. Li, ``Content Centric Peer Data Sharing in Pervasive Edge Computing Environments,'' IEEE ICDCS 2017.
- Y. Huang, X. Song, F. Ye, Y. Yang and X. Li, ``Fair Caching Algorithms for Peer Data Sharing in Pervasive Edge Computing Environments,'' IEEE Transactions on Mobile Computing, pp. 852-864, 2020.
- Y. Huang, J. Zhang, J. Duan, B. Xiao, F. Ye and Y. Yang, Resource Allocation and Consensus on Edge Blockchain in Pervasive Edge Computing Environments. IEEE ICDCS 2019, to appear in IEEE Transactions on Mobile Computing.
- X. Yan, F. Ye, Y. Yang and X. Deng, ``Autonomous Compensation Game to Facilitate Peer Data Exchange in Crowdsensing,'' to appear in IEEE Transactions on Cloud Computing.

Thank you!

Yuanyuan Yang
www.ece.stonybrook.edu/~yang
yuanyuan.yang@stonybrook.edu